Esko Kantola

Salkunhallintaohjelmiston kaupparaportoinnin automatisointi

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

29.11.2015

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Esko Kantola Salkunhallintaohjelmiston kaupparaportoinnin automatisointi 32 sivua 29.11.2015
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaajat	Lehtori Simo Silander Toimitusjohtaja Juha Lehtonen

Insinöörityön tavoitteena oli suunnitella ja toteuttaa salkunhallintaohjelmistoon FA Portfolio lisäominaisuus, joka mahdollistaa automatisoidusti tapahtuvan kaupparaportoinnin FA Portfolio -sovelluksesta Finanssivalvonnan käyttämään TYVI-palveluun. Työn lähtökohtana oli tuoda olemassa olevan manuaalisia vaiheita sisältävän kaupparaportoinnin rinnalle vaihtoehto, jossa kaupparaportoinnin eri vaiheet tapahtuisivat ohjelmallisesti ilman sovelluksen käyttäjien panosta.

Työn teoriaosuudessa käsiteltiin yleisellä tasolla salkunhallintaohjelmistoa FA Portfolio ja mistä tiedoista Finanssivalvonnalle raportoitava kauppatapahtuma koostuu. Automatisoinnin ohjelmallistaminen toteutettiin käyttämällä FA Portfolio -sovelluksen tukemaa Apache Camel -sovelluskehystä, joka on toteutettu erilaisten järjestelmien ja tietotyyppien integrointiin. Työssä käytiin vaihe vaiheelta läpi eri integraatiot automatisoidun kaupparaportoinnin aikaansaamiseksi.

Lopputuloksena syntyi FA Portfolio -sovellukselle lisäominaisuus, joka mahdollistaa sijoituspalveluyritysten hoitaa kaupparaportointi täysin automatisoidusti.

Avainsanat	Salkunhallintaohjelmistot, raportointi, integraatiot, Apache Camel

Author	Esko Kantola
Title	Automating Transaction Reporting of Portfolio Management software
Number of Pages	32 pages
Date	29 November 2015
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructors	Simo Silander, Senior Lecturer Juha Lehtonen, Managing Director

The purpose of this thesis was to design and implement a new addition for the portfolio management software FA Portfolio. The new addition would enable an automated way to send transaction reports from the FA Portfolio to the TYVI service used by Financial Supervisory Authority. The core idea of the study was to introduce an alternative way to do transaction reporting where different phases of transaction reporting are handled programmatically and would not require any manual input of the users.

The theoretical part focuses on the portfolio management software FA Portfolio in general and introduces what kind of information a transaction contains that needs to be reported to Financial Supervisory Authority. Automating the transaction reporting was implemented by using the Apache Camel framework supported by FA Portfolio. The Apache Camel framework is built for managing the integrations between different kind of systems and data types. The practical part of the thesis introduces phase-by-phase the different integrations that produce an automated way to do transaction reporting.

As a result of the thesis, a new addition that enables investment firms to do transaction reporting fully automatically was built for the FA Portfolio software.

Keywords	Portfolio management software, reporting, integrations, Apache Camel

Sisällys

Lyhenteet

TYVI	*Tietovirrat yritysten ja viranomaisten välillä*. Järjestelmä tarjoaa yrityksille yhdenmukaisen tavan ilmoittaa tietoja sähköisesti viranomaisille ja viranomaistehtäviä hoitaville tahoille.
XML	*Extensible Markup Language.* Rakenteellinen tietojen kuvauskieli, joka jäsentää laajoja tietomassoja selkeästi.
AJAX	*Asynchronous JavaScript And XML.* Web-selaimen ja palvelinpään tiedonvälitykseen asynkronisesti mahdollistava tekniikka.
CSV	*Comma-separted values.* Tekstiä sisältävä tiedostomuoto, jossa tieto on tallennettu taulukkorakenteeseen käyttäen sarakkeiden erotinmerkkinä pilkkua.
ISO	*International Organization for Standardization.* Kansainvälinen standardoimisjärjestö.
BIC	*Business Identifier Code.* ISO:n standardoima yksilöllinen tunniste finanssialan yhteisöille ja yrityksille.
FTP	*File Transfer Protocol.* Asiakas- ja palvelintietokoneiden välinen tiedostonsiirtomenetelmä.
MEPs	*Message Exchange Patterns.* Viestinvälitysmalli, joka tyypillisesti jakautuu joko yksisuuntaiseen viestinvälitykseen tai kutsu−vastaus-tyyppisen viestinvälitykseen.
URI	*Uniform Resource Identifier.* Merkkijono yhdenmukaisen resurssin tunnistamiseksi.
DSL	*Domain-specific Language.* Sovelluskohtainen ja -riippuvainen täsmäkieli.
XSLT	*Extensible Stylesheet Language Transformations.* Muunnoskieli, jota käytetään XML-dokumentin tietosisällön muokkaukseen esimerkiksi html-tiedostoksi.

SMTP	*Simple Mail Transfer Protocol.* Sähköpostipalvelinten välillä tapahtuvaan viestien välittämiseen käytettävä protokolla.

1 Johdanto

Suomessa toimiva valvontaviranomainen Finanssivalvonta valvoo, että kotimaiset pankit, vakuutus- ja eläkeyhtiöt, sijoitusrahastot ja sijoituspalveluyritykset harjoittavat toimintaansa terveellä pohjalla siten, että valvottavilla on pääomia toiminnasta aiheutuvien riskien ja tappioiden kattamiseen ja että valvottavat eivät ota kohtuuttomia riskejä. Valvontaa tehdään muun muassa erilaisin tarkastuksin sekä analysoimalla valvottavilta tahoilta säännöllisesti saatavia raportteja. Yhtenä valvontaraporttina toimii kaupparaportointi, jossa sijoituspalvelutoimintaa harjoittava yritys raportoi asiakkaidensa lukuun rahoitusvälineillä tehdyistä kaupoista. [1]

FA Solutions Oy tarjoaa sijoituspalveluyrityksille, jotka harjoittavat sijoitusneuvontaa ja omaisuudenhoitoa, salkunhallintaohjelmistoa nimeltä FA Portfolio. Salkunhallintaohjelmiston yhtenä ominaisuutena on määrämuotoisen kaupparaportin tuottaminen sovellukseen kirjatuista kauppatapahtumista. Raportoitavan kauppatapahtuman kohteena on rahoitusvälineinstrumentit, kuten osakkeet ja joukkovelkakirjalainat, joita sijoituspalveluyritys on ostanut tai myynyt asiakkaansa lukuun.

FA Portfolio -sovellusta käyttävien sijoituspalveluyritysten on tällä hetkellä mahdollista raportoida tehdyt kaupat tallentamalla sovellusta käyttävän henkilön tietokoneelle sovelluksen generoima XML-dokumentti, joka kuvaa tehtyjä kauppoja. Tämän jälkeen käyttäjän tulee siirtää XML-tiedosto Finanssivalvonnalle analysointia varten. Raportointi tapahtuu TYVI-palvelun kautta, joka on yrityksille suunnattu yhdenmukainen järjestelmä sähköiselle raportoinnille eri viranomaisille. FA Portfolio -sovelluksen käyttäjät lataavat tehdyt kaupat sisältävän XML-dokumentin TYVI-palveluun selainversion kautta.

Edellä mainittu tapa raportoida kauppoja vaatii FA Portfolio -sovelluksen käyttäjältä manuaalista työtä, joka sisältää tiedostolataamisen ja sen jälkeisen seurannan TYVI-palveluun raportoitujen kauppojen tilasta, jotka voivat olla tilassa odottaa, hyväksytty tai hylätty.

Insinöörityön tavoitteena on jatkokehittää FA Portfolio -salkunhallintaohjelmiston kaupparaportointi olemaan mahdollisimman automatisoitu. Automatisoidun raportoinnin tulisi hallita kaupparaportin välittäminen TYVI-palveluun sekä TYVI-palvelusta saatavan

vastausviestin tulkitsemisen. Vastausviesti sisältää status-tiedon jokaisesta raportoitavaksi lähetetystä kaupasta. Statuksena on joko hyväksytty tai hylätty. FA Portfolio -sovelluksen käyttäjää tulisi informoida raportoitujen kauppatapahtumien osalta, mitkä kaupat raportoitiin hyväksytysti ja mitkä hylättiin. Hylättyjen kauppatapahtumien osalta hylkäyksen syy tulee myös ilmetä sähköpostiviestissä.

Työssä tullaan käyttämään FA Portfolio -sovelluksessa käytettävää integraatioihin tarkoitettua Apache Camel -sovelluskehystä, jonka avulla tullaan muodostamaan XML-pohjainen kaupparaporttitiedosto, sen välittäminen TYVI-palvelun FTP-palvelimelle, TYVI-palvelusta ladatun vastausviestin tulkinta ja FA Portfolio -sovelluksen käyttäjää informoiminen sähköpostitse raportoinnin lopputuloksesta hyväksytyt ja hylätyt kauppatapahtumat eriteltyinä.

Kaupparaportoinnin jatkokehityksen onnistuessa on FA Portfolio -sovelluksen käyttäjillä mahdollisuus hankkia sovellukseen lisäpalveluna kaupparaportoinnin automatisoitu versio, jossa sovellus ajastetusti tuottaa kaupparaportin sovelluksessa raportoitaviksi merkityistä kaupoista, välittää raportin Finanssivalvonnalle ja merkitsee kauppatapahtumat raportoiduiksi.

Insinöörityön toimeksiantaja FA Solutions Oy on vuonna 1999 perustettu finanssialan toimijoille IT-ratkaisuja toimittava yritys, jonka ratkaisut tukevat sijoitussalkkujen hallinnointia, sijoitusten analysointia ja raportointia sekä finanssituotteiden myyntiä.

2 FA Portfolio

2.1 Salkunhallintajärjestelmät

Sijoitussalkkujen hallintaan toteutetut järjestelmät tarjoavat alustan, jossa käyttäjät voivat pitää kirjaa omistamistaan tai hallinnoimistaan erityyppisistä sijoitusinstrumenteista. Perinteisiä sijoitusinstrumenttityyppejä ovat mm. julkisen kaupankäynnin kohteena olevat pörssinoteeratut osakkeet, yritysten tai valtioiden liikkeelle laskemat joukkovelkakirjalainat tai rahastoyhtiöiden hallinnoimat osake- ja korkorahastot. Tyypillisesti salkunhallintajärjestelmät tarjoavat käyttäjälleen informaationa, paljonko tiettyä sijoitusinstrumenttia omistetaan, mikä on sijoitusinstrumentin hankinta-arvo ja

nykyinen markkina-arvo sekä edellä mainittujen arvojen erotus, joka kuvaa, miten sijoitusinstrumentti on tuottanut.

Salkunhallintajärjestelmän käyttäjiä voivat olla esimerkiksi yksittäiset sijoittajat, jotka haluavat keskitetyn paikan, jonne kirjata sijoitusomistuksensa ja josta he saavat kokonaiskuvan sijoitusvarallisuudestaan, tai sijoituspalveluyritykset, jotka tarjoavat varainhoitopalveluita asiakkailleen pitäen kirjaa asiakkaiden lukuun tehdyistä sijoitusinstrumenttien hankinnoista salkunhallintajärjestelmässä.

Salkunhallintajärjestelmien ominaisuuksiin kuuluu myös erilaisten raporttien tuottaminen. Raportit voivat käsitellä mm. sijoitusten hankintojen ja myyntien tapahtumalistausta, sijoitusten nykytilan hahmottamista graafisin keinoin – esimerkiksi miten sijoitukset ovat jakautuneet eri sijoitusinstrumenttilajeittain ja -sektoreittain – sekä eri viranomaistahojen edellyttämiä sijoitusten seurantaan liittyviä valvontaraportteja.

2.2 Yleistä sovelluksesta

FA Portfolio on FA Solutionsin tuotteistama Java-pohjainen sovellus, jota ajetaan Web-sovelluksena Apache Tomcat -palvelimella. Sovellusta on mahdollista ajaa servlet-versiona sekä JSR 286 -standardin mukaisena portlet-versiona osana portaalin tarjoamaa Container-alustaa. Edellä mainituista varsinkin portlet-versio on yleisimmin käytetty versio FA Portfolio -sovelluksesta johtuen portaalialustan tuomista muista ominaisuuksista, joihin kuuluvat muun muassa käyttäjähallinta sekä valmistyökalut sivustojen ja sisällön tuottamiseen sekä hallintaan.

Salkunhallintajärjestelmien tapaan FA Portfoliossa käsiteltävä tieto koostuu ylläpidettävistä sijoitusinstrumenteista eli arvopapereista, sijoitussalkuista ja niihin linkitetyistä asiakaskontakteista sekä sijoitussalkkuihin kohdistuvista kauppatapahtumista.

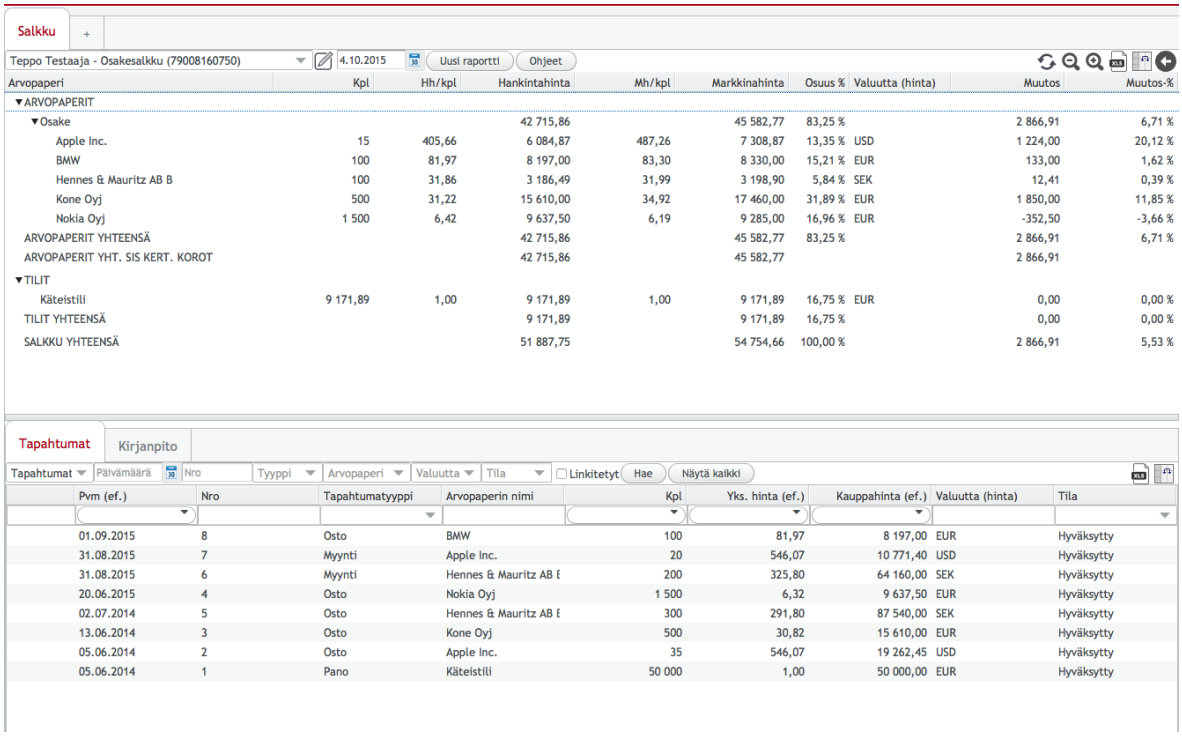
Arvopaperin tiedoissa ylläpidetään perustietoja kuten arvopaperin nimi, yksilöllinen arvopaperikoodi, valuutta, jolla arvopaperin kaupankäynti tapahtuu, arvopaperin päiväkohtaisia päätöskurssien markkinahintahistoriaa sekä arvopaperin tyyppitietoa, joita ovat esimerkiksi osake, joukkovelkakirjalaina tai rahasto.

Sijoitussalkkujen osalta ylläpidettävää tietoa ovat salkun nimi, järjestelmäkohtainen yksilöllinen salkkutunnus, sijoitussalkun raportoinnissa käytettävä valuutta ja asiakaskontakti, jonka omistuksessa sijoitussalkku ja siellä olevat arvopaperit ovat.

Asiakaskontaktin tietoihin kirjataan asiakkaan nimi- ja yhteystiedot, yksilöllinen tunnus, joka tavallisesti on joko henkilötunnus tai y-tunnus sekä kontaktin juridinen muoto esimerkiksi henkilöasiakas tai yritysasiakas.

FA Portfolion käyttöliittymä on toteutettu käyttäen avoimen lähdekoodin Vaadin-ohjelmointirajapintaa, joka tarjoaa käyttöliittymäkomponenttien muodossa kokonaisuuden, jossa palvelinpään Back-End-toiminnallisuus on kytköksissä interaktiiviseen asiakaspään käyttöliittymään AJAX-tekniikan avulla [2].

Kuvassa on 1 esiteltynä sovelluksen Yhteenveto-näkymä, jossa tarkastelun kohteena ovat asiakaskontaktin sijoitussalkun positionäkymän sisältö ja listaus sijoitussalkkuun kohdistuneista kauppatapahtumista.



Kuva 1. FA Portfolio -sovelluksen Yhteenveto-näkymä.

2.3 Kauppatapahtuma FA Portfoliossa

FA Portfolio -sovelluksessa kauppatapahtuman oleellisimmat tiedot ovat kauppatapahtuman tyyppi kuten osto tai myynti, tapahtumapäivä, sijoitussalkku, johon kauppatapahtuma on kohdistunut, kaupankäynnin kohteena ollut arvopaperi sekä kappalemäärä, joka kertoo, kuinka moneen yksikköön arvopaperia kauppatapahtuma on kohdistunut ja mihin yksikköhintaan. Edellä manittujen kappalemäärän ja yksikköhinnan perusteella sovellus laskee kauppatapahtuman kokonaissumman.

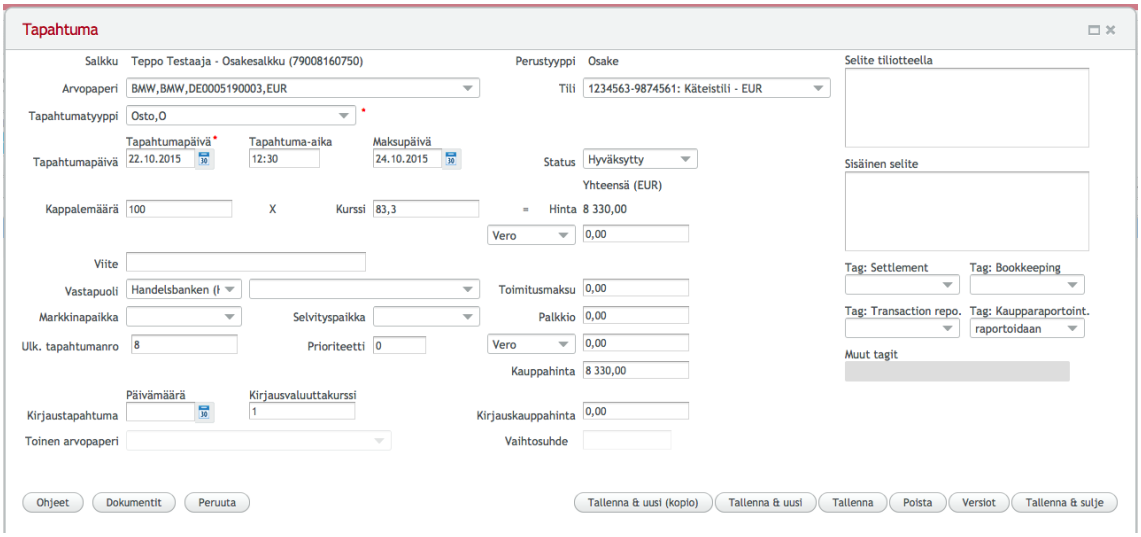
Edellä mainittujen kauppatapahtumaan liittyvien vähimmäistietojen perusteella voidaan esimerkiksi ostotyyppisen kauppatapahtuman tiedoista luoda FA Portfoliossa asiakaskontaktin sijoitussalkkuun arvopaperille omistuspositio, josta ilmenee, kuinka monta kappaletta arvopaperia omistetaan, mikä on ollut omistusposition kokonaishankintahinta ja mikä on sen nykyinen kokonaisarvo perustuen arvopaperin nykyiseen markkinahintaan.

Vähimmäistietojen lisäksi kauppatapahtumalle on mahdollista tallentaa tarvittaessa monipuolisempaa tietoa. Tällaista tietoa on muun muassa kauppatapahtumaan liittyvä palkkio, sijoitussalkun tietoihin määritetty käteistili, jota on käytetty kauppatapahtuman yhteydessä ja jossa käteisvirtaama näkyy joko negatiivisena tai positiivisena riippuen kauppatapahtuman tyypistä, kauppatapahtuman vastapuoli eli kenen kanssa kauppa on käyty sekä kauppatapahtuman tagi-tieto, joka on kauppatapahtumiin liittyvää dynaamisesti muodostettavaa metatietoa, jonka perusteella voidaan tehdä monipuolisia kauppatapahtumahakuja sovelluksessa.

Kauppatapahtumia voidaan syöttää sovellukseen käyttäen käyttöliittymän tarjoamaa tapahtuman tallennusominaisuutta. Graafiselta käyttöliittymältä kauppatapahtumia luodaan yksi kerrallaan, tämänkaltainen toimintatapa on toimiva sovelluksen käyttäjille, joilla tallennettavia kauppatapahtumia ei ole useita per päivä. Kuvassa 2 on esiteltynä sovelluksen kauppatapahtuman syöttöruutu.

Sovelluksen käyttäjillä, kuten omaisuudenhoitopalveluita tarjoavilla sijoituspalveluyrityksillä, on yleensä tarpeen tallentaa sovelluksessa useita kauppatapahtumia asiakkaidensa sijoitussalkkuihin per päivä. Kauppatapahtumien massatallennusta varten on sovellukseen kehitetty kauppatapahtumien import-ominaisuus. Kauppatapahtumien massatallennuksessa sovellukseen tuodaan

käsiteltäväksi määrämuotoinen CSV-tiedosto, jossa kauppatapahtumat ovat rivitietona ja kauppatapahtumaan liittyvät tiedot ovat erotinmerkillä eriteltyinä omissa sarakkeissaan.



Kuva 2. Ostotyyppisen tapahtuman syöttöruutu vähimmäistiedoilla täytettynä.

3 Kaupparaportointi FA Portfolio -sovelluksessa

3.1 Kaupparaportoitava kauppatapahtuma

Kaupparaportointi on FA Portfolio -sovelluksen lisäominaisuus, jonka käyttöönoton yhteydessä käydään sovelluksen asiakaskäyttäjän kanssa läpi määrittelyvaihe, jossa käsitellään, minkälaisille kauppatapahtumille he ovat raportointivelvollisia Finanssivalvonnalle. Määrittelyvaiheen jälkeen FA Portfolio -sovellukseen toteutetaan säännöt, joiden perusteella sovellukseen tallennettaville tapahtumille lisätään tagi-tieto ”Kaupparaportointi - raportoidaan”. Sääntöjen toteuttamiseen ja tapahtuman tagi-tiedon lisäämiseen käytetään Javalla toteutetun Drools-projektin Rule-skriptikieltä [3]. Rule-skripti suoritetaan aina, kun FA Portfolio -sovelluksessa tallennetaan kauppatapahtumaa. Skriptissä tarkastellaan tallennettavan tapahtuman tietoja ja verrataan niitä ehtoihin, jotka ovat määritetty kaupparaportoitavalle tapahtumalle. Tyypillisesti määritettyjä ehtoja ovat: kauppatapahtuman tyypin on oltava osto tai myynti, kauppatapahtuma on hyväksytty sekä kauppatapahtumalle on määritetty vastapuoli.

Tallennettavan kauppatapahtuman sisältäessä vaadittavat tiedot Rule-skripti lisää tapahtumalle tagin ”Kaupparaportointi - raportoidaan”.

Kaupparaportti muodostetaan päivittäin kauppatapahtumista, joilta löytyy tagi-tieto ”Kaupparaportointi - raportoidaan”. Kaupparaporttia varten haetaan tapahtumalta, sijoitussalkusta ja sijoitussalkun omistajasta Taulukko 1:n mukaiset tiedot.

Taulukko 1. Kaupparaportoinnin kauppatapahtumakohtaiset kentät.

TransactionReferenceNumber	Kauppatapahtuman yksilöllinen viite. Muodostetaan syntaksilla <sijiotussalkun_tunnus>-<tapahtumanumero>
TradingTimestamp	Tapahtuman päivämäärä ja kellonaika
BuySellIndicator	Arvo B = osto, arvo S = myynti
Instrument	Kaupankäyntikohteena olleen instrumentin ISIN-koodi
UnitPrice	Tapahtuman yksikköhinta kaupankäyntivaluutassa
PriceNotation	Kaupankäyntivaluutta ilmoitettuna ISO 4217 -standardin mukaisella valuuttakoodilla
Quantity	Tapahtuman kokonaiskappalemäärä
CounterParty	Kaupan vastapuolena toimineen liikelaitoksen BIC-tunnus
Client CodeType ”I”	Sijoitussalkun omistajan järjestelmäkohtainen, yksilöllinen tunnus. Arvo haetaan sovelluksen kontakti-tietokantataulun asiakaan id-kentästä.
ClientName	Sijoitussalkun omistajan nimi
ClientIdentificationLocal	Sijoitussalkun omistajan henkilö- tai Y-tunnus
ClientStreet	Sijoitussalkun omistajan katuosoite
ClientZipCode	Sijoitussalkun omistajan postinumero
ClientCity	Sijoitussalkun omistajan postitoimipaikka
ClientCountry	Sijoitussalkun omistajan asuinmaa

3.2 Käyttöönotto

Kaupparaportoinnin käyttöönotto aloitetaan luvussa 3.1 esitetyn mukaisesti määrittelyvaiheella, jonka perusteella sijoituspalveluyrityksen FA Portfolio -sovellukseen

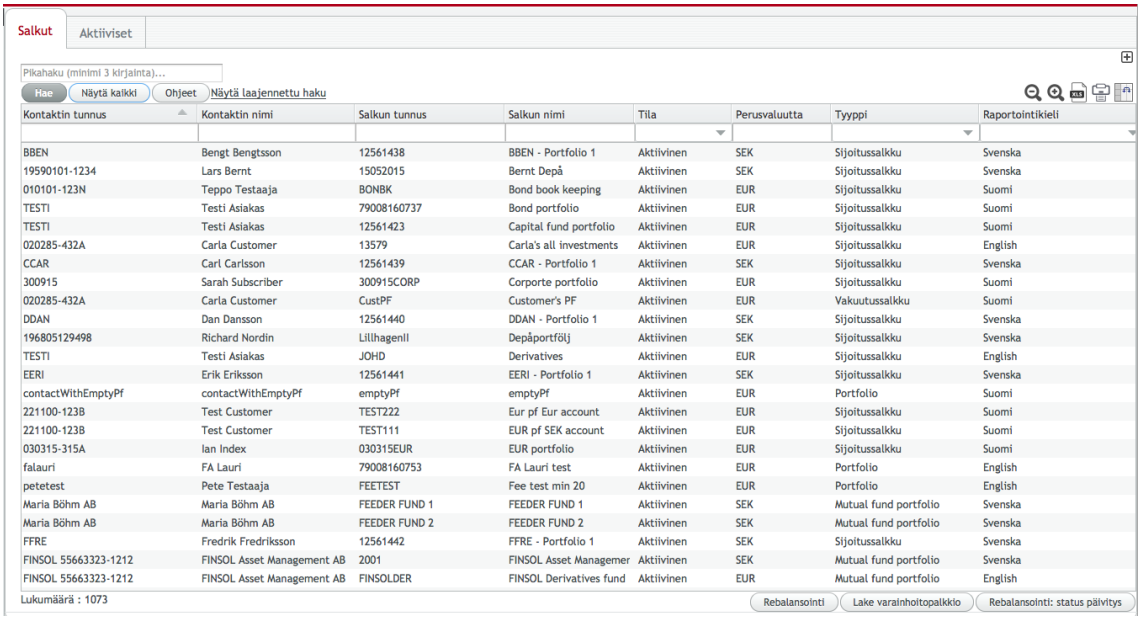
konfiguroidaan Drools Rule -skripti, joka tulee lisäämään sovelluksessa tallennettaville tapahtumille tagin ”Kaupparaportointi – raportoidaan”, kun tallennettavan tapahtuman tiedot vastaavat kaupparaportoitavaksi määritellyn tapahtuman tietoja.

Kauppatapahtumien tagi-konfiguroinnin jälkeen tulee FA Portfolio -sovellukseen konfiguroida vielä kaupparaportointia varten raportoijana toimivan sijoituspalveluita tarjoavan yrityksen ISO 9362 -standardin mukainen BIC-tunniste. Tunnistetta tullaan käyttämään XML-muotoisen kaupparaportin tiedostonimessä sekä raportilla elementtien ReportingFirm ja TechnicalReportingFirm arvoina, jotka merkitsevät lähetettävän kaupparaportin raportoijaksi nimenomaisen sijoituspalveluyrityksen. [4]

Kaupparaportointiin liittyvien konfigurointien ollessa valmiina, viimeisenä vaiheena kaupparaportoinnin käyttöönottoon liittyen on asentaa kaupparaporttitiedoston tuottava Groovy-ohjelmaskripti. Kaupparaporttitiedoston tuottava ohjelmaskripti on alkuvaiheen asiakaskohtaisen konfiguroinnin jälkeen täysin geneerinen ja täten tuotteistamisen perusarvot omaava ohjelmaskripti. Kaupparaportin sisältö tuotetaan skriptissä käyttäen groovy.xml-paketin luokkaa MarkupBuilder, joka on yleisesti käytetty ja hyvin dokumentoitu esimerkkien avulla Groovyn Internet-sivuilla [5]. XML-dokumentin rakenne ja tuotettavat elementit noudattavat Finanssivalvonnan dokumentissa määriteltyä XML-skeemaa [4].

FA Portfolio -sovellukseen on toteutettu oma Groovy-tulkki, joka tarjoaa tulkin kautta suoritettaville Groovy-skripteille FA Portfolio -sovelluksen palvelurajapinnat mahdollistaen näin ohjelmaskripteille pääsyn muun muassa lukemaan ja kirjoittamaan tietoa FA Portfolio -sovelluksen tietokantadataan. Tulkille on toteutettu sovelluskontrollerin avulla mahdollisuus vastaanottaa suoritettaville Groovy-skripteille käyttöliittymällä haettuna olevia salkunhallintajärjestelmän entiteettejä kuten asiakaskontakteja, sijoitussalkkuja ja kauppatapahtumia. Groovy-tulkissa suoritettavalle skriptiohjelmalle välitetään skriptikoodissa sovelluskohtainen nimiavaruustieto kuten contacts, portfolios tai transactions, joka huomioidaan FA Portolio -sovelluksen kontrollerilla siten, että käyttöliittymällä näytetään nimiavaruutta vastaavan näkymän, Kontaktit, Salkut tai Tapahtumat alareunassa skriptin ajamiseen liittyvä toimintonappi. Suoritettavalle Groovy-skriptille on mahdollista antaa skriptikoodissa FA Portfolio -tulkin ymmärtämän syntaksin mukainen nimi toimintonapille, joka näkyy myös käyttöliittymällä. Kuvassa 3 on esitettynä sovelluksen Salkut-näkymä, jossa alareunan toimintonapeilla ”Rebalansointi”, ”Laske varainhoitopalkkio” ja ”Rebalansointi: status päivitys” voidaan

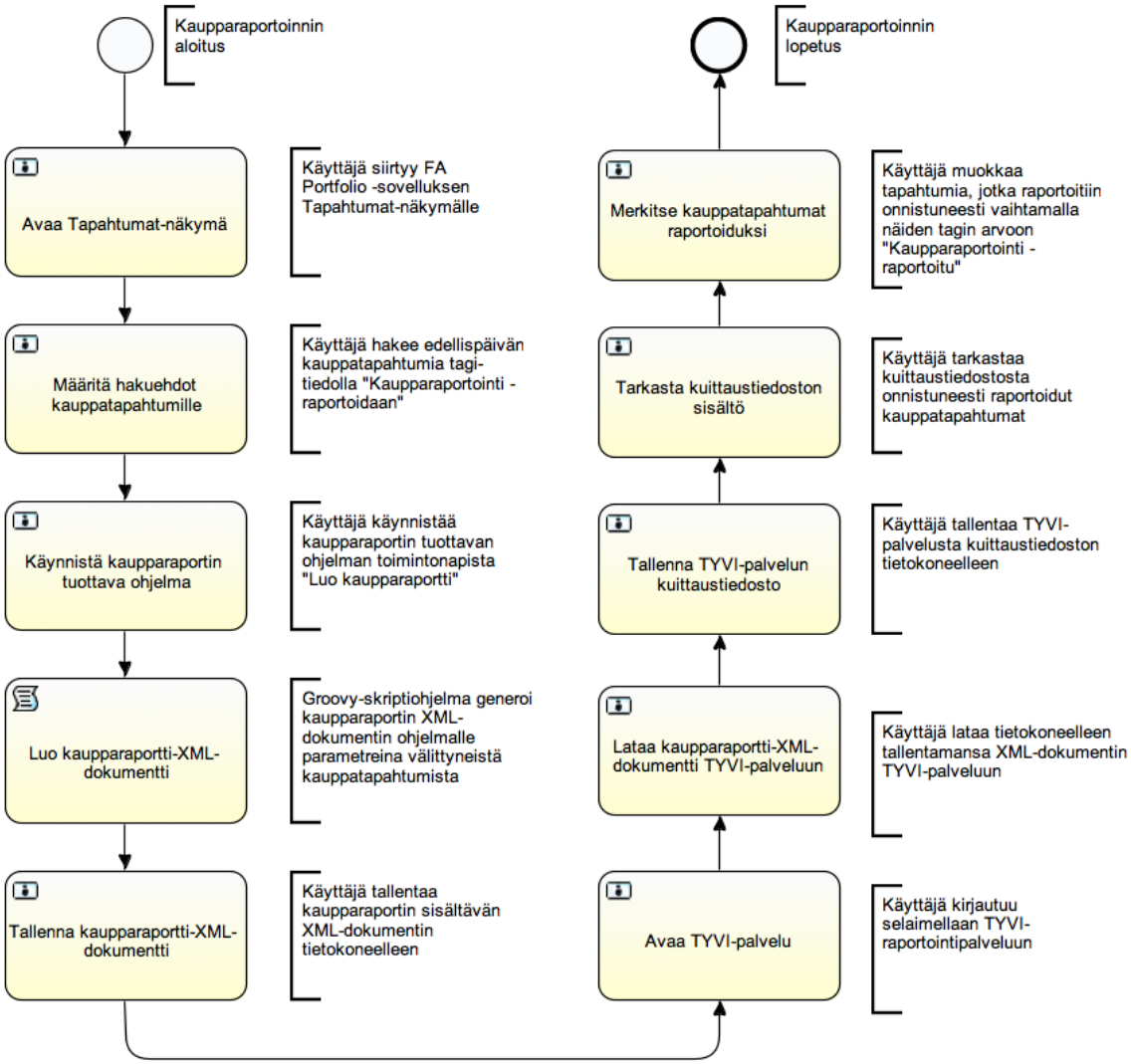
ajaa Groovy-tulkin kautta toimintonappeihin liittyviä skriptiohjelmia, jotka ottavat parametrikseen käyttöliittymällä haettuna olevat sijoitussalkut. Tulkin ja suoritettavan Groovy-skriptin avulla on FA Portfolio -sovellukseen mahdollista saada tuotettua dynaamisesti asiakaskohtaisia lisäominaisuuksia, kuten esimerkiksi varainhoitopalkkioiden laskeminen Salkut-näkymässä haetuille salkuille tai sähköpostitiedotteen lähettäminen Kontaktit-näkymässä haetuille asiakaskontakteille.



Kuva 3. Salkut-näkymän sijoitussalkkulistaus ja toimintonapit salkku-entiteettejä parametrina ottavien skriptiohjelmien suorittamiseen.

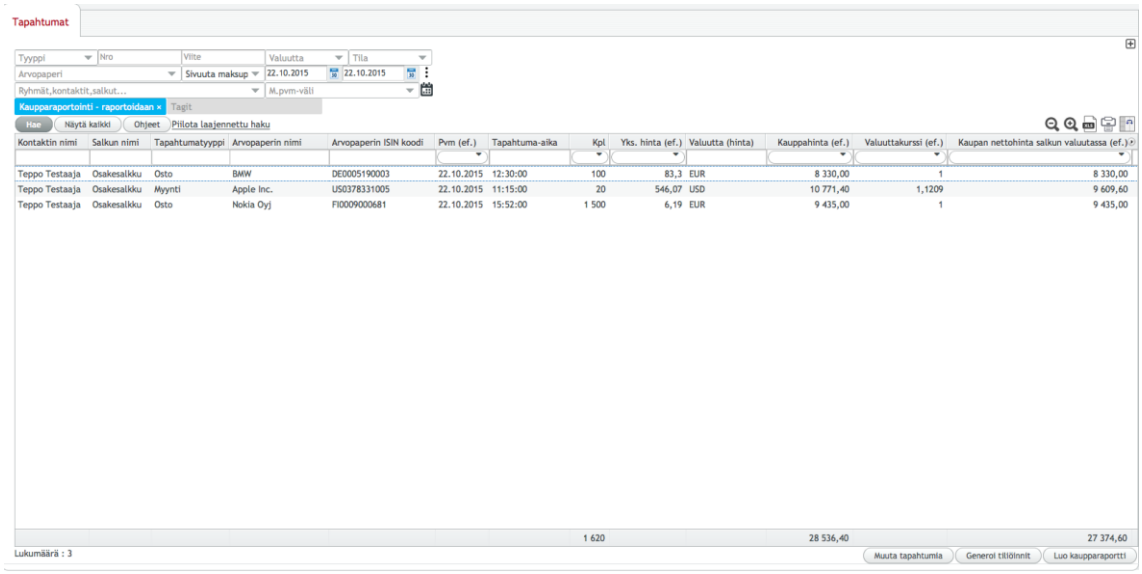
3.3 Manuaalisen raportoinnin vaiheet

Manuaalinen kaupparaportointi koostuu FA Portfolio -sovelluksen käyttäjän vaiheittain tekemistä tehtävistä. Kuvassa 4 on esitelty päivittäisen kaupparaportoinnin eri vaiheet, jotka on eroteltu käyttäjän tekemiin tehtäviin ja ohjelmallisesti tapahtuvaan tehtävään.



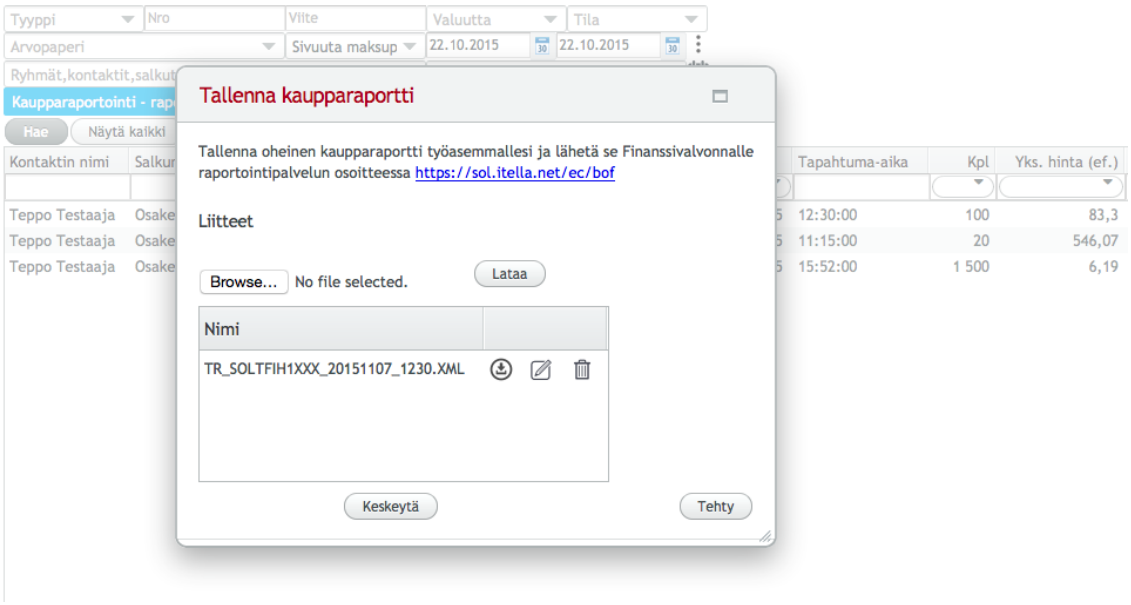
Kuva 4. Manuaalisen kaupparaportoinnin vaiheet.

Arvopaperimarkkinalain 14 luvun 7 §:n 1 ja 5 momentin mukaan sijoituspalveluyrityksen tulee raportoida kauppatapahtuma viimeistään seuraavana pankkipäivänä. Tämä edellyttää FA Portfolio -sovelluksen käyttäjiltä päivittäin tapahtuvaa aamurutiinia, jossa sovelluksen Tapahtumat-näkymällä haetaan edellisen päivän kauppoja, joilla löytyy tagi ”Kaupparaportointi – raportoidaan”. FA Portfolio -sovelluksen Tapahtumat-näkymällä tehty päivämäärä- ja tagi-pohjainen haku on esitelty kuvassa 5. Haun löytäessä tapahtumia, kaupparaporttitiedoston generointi käynnistetään toimintonapista ”Luo kaupparaportti”.



Kuva 5. Tapahtumat-näkymällä haettu tietyn päivän tapahtumat tagilla ”Kaupparaportointi - raportoidaan”.

Skriptiohjelma saa edellisessä luvussa esitetyn mukaisesti Tapahtumat-näkymällä haettuna olevat tapahtuma-entiteetit parametrina, jotka sisältävät tietona taulukko 1:n mukaiset tiedot tapahtumasta, sijoitussalkusta sekä sijoitussalkun omistajasta XML-dokumentin sisällön muodostamiseksi. Kaupparaportin luonnin jälkeen FA Portfolio -sovellus tarjoaa käyttäjälle kaupparaportin tiedoston ladattavaksi käyttäjän työkoneelle, kyseinen latausikkuna on esitetty kuvassa 6. Käyttäjälle näytettävä latausikkuna on FA Portfolio -sovellukseen toteutettu käyttöliittymäkomponentti, jonka avulla käyttöliittymälle on mahdollista tarjota erityyppisiä tiedostoformaatteja ladattavaksi. Käyttöliittymäkomponentti vastaanottaa konstruktorilleen parametreina tiedoston binääridatan, tiedostonimen, ikkunaotsikon ja saatetekstin.



Kuva 6. Generoidun kaupparaportti-XML-dokumentin tallennusikkuna FA Portfolio -ovelluksen latausikkunakomponentissa.

Skriptiohjelman lopputuloksena tuottama XML-dokumentti koostuu kaupparaportin skeeman määrittelyn mukaisesti raportoijan tietoelementeistä ReportingFirm ja TechnicalReportingFirm sekä itse kauppatapahtumista, joiden tiedot on sisällytetty elementteihin Transaction. Kuvassa 7 on esitelty XML-dokumentin alkua, jossa kerrotaan raportoijana toimivan sijoituspalveluyrityksen BIC-tunniste ja tiedot esimerkkiasiakkaasta Teppo Testaajasta sekä tehdystä osakekaupasta, jossa myytiin Tepon hallussa ollutta Apple Inc. -arvopaperia.

```
<?xml version='1.0' encoding='UTF-8'?>
<ns:report xmlns:ns='http://schemas.fi.se/TRS/InTrans' Version='3.00'>
  <TechnicalReportingFirm Identification='SOLTFIH1XXX' />
  <ReportingFirm Identification='SOLTFIH1XXX'>
    <Transaction>
      <TransactionReferenceNumber>79008160750-7</TransactionReferenceNumber>
      <TradingTimestamp>2015-10-22T11:15:00.000+03:00</TradingTimestamp>
      <BuySellIndicator>S</BuySellIndicator>
      <TradingCapacity>P</TradingCapacity>
      <Instrument>US0378331005</Instrument>
      <UnitPrice>
        <PriceCurrency>546.07</PriceCurrency>
      </UnitPrice>
      <PriceNotation>USD</PriceNotation>
      <Quantity>20.0</Quantity>
      <CounterParty CodeType='B'>NDEAFIHHXXX</CounterParty>
      <Venue CodeType='O'>XOFF</Venue>
      <Client CodeType='I'>144</Client>
      <ClientName>Teppo Testaaja</ClientName>
      <ClientIdentificationLocal>010101-123N</ClientIdentificationLocal>
      <ClientStreet>Testaajankatu 42 A 14</ClientStreet>
      <ClientZipCode>00100</ClientZipCode>
      <ClientCity>Helsinki</ClientCity>
      <ClientCountry>FI</ClientCountry>
    </Transaction>
```

Kuva 7. Kaupparaportti-XML-dokumentin sisältöä. Esitettynä raportoijan ja esimerkkitapahtuman Apple Inc. -osakkeen myyntiin liittyvät tiedot.

Manuaalisen kaupparaportoinnin viimeisinä vaiheina on siirtää XML-dokumentti TYVI-palveluun, joka prosessoi raportin tavallisesti noin kymmenessä minuutissa tuottaen lopputuloksena XML-pohjaisen vastaustiedoston. Vastaustiedostosta selviää, mitkä raportoidut tapahtumat raportoitiin onnistuneesti ja mitkä tapahtumat hylättiin. Käyttäjän tulisi tarkistaa vastaustiedostolta onnistuneesti raportoitujen kauppojen TransactionReferenceNumber ja käydä vaihtamassa kyseisille tapahtumille FA Portfolio -sovelluksessa kaupparaportointiin liittyvälle tagille arvo ”Kaupparaportointi - raportoitu”. Hylättyjen kauppatapahtumien osalta vastaustiedostolla on hylkäyksen tarkennekoodi, joka useimmiten vaatii jatkotoimenpiteenä kauppatapahtuman muokkausta FA Portfolio -sovelluksessa ja kaupparaportin uudelleen lähettämistä muokattujen tapahtumien osalta. Uudelleen raportoitavien, korjattujen tapahtumien kanssa toimitaan noudattamalla samoja vaiheita kuin kyseisen raportointipäivän ensikerran raportoinnissa, josta tagi-rajoitin ”Kaupparaportointi – raportoidaan” jättää huomioimatta onnistuneesti raportoidut tapahtumat, joille käyttäjä on käynyt vaihtamassa tagin arvoon ”Kaupparaportointi – raportoitu”.

3.4 Raportoinnin eri vaiheiden automatisointi

Manuaalisen kaupparaportoinnin eri vaiheet koostuvat neljästä selkeästä kokonaisuudesta, jotka voidaan automatisoida ohjelmallisesti: raportin muodostus, raportin lähettäminen TYVI-palveluun, tiedoston noutaminen ja tulkitseminen sekä onnistuneesti raportoitujen tapahtumien tagi-tiedon vaihtaminen arvoon ”Kaupparaportointi - raportoitu”.

3.4.1 Raporttitiedoston luominen

Tapahtumat, joilla on FA Portfolio -sovelluksessa tagi ”Kaupparaportointi - raportoidaan” tulee raportoida Finanssivalvonnalle viimeistään seuraavana pankkipäivänä. Edellä mainitut ehdot toimivat täten hakukriteerinä sovelluksen raportoitavia tapahtumia haettaessa. Haku tulee tehdä kerran vuorokaudessa pankkipäiviä seuraavina viikonpäivinä eli tiistaista lauantaihin, jolloin hakuehdoksi muodostuu lauseke ”tapahtumapäivän tulee olla kuluvaa päivää edeltävä päivä ja tapahtumalla tulee olla tagi ”Kaupparaportointi - raportoidaan””. Ohjelmallisesti haun aloitus on mahdollista tehdä ajastimen avulla, jolle ajastimen syntaksin mukaisesti pystytään määrittämään toistuva aloitusajankohta kellonajan ja viikonpäivän mukaan. Raporttitiedoston luominen voidaan manuaalisen tapahtumahaun ja ”Luo kaupparaportti” -napin painamisen sijaan siis käynnistää ohjelmallisesti ajastimen avulla. Ajastimella käynnistetyn ohjelman tulisi hakea hakukriteerin omaavat tapahtuma-entiteetit ja välittää ne parametrina kaupparaporttitiedoston muodostavalle geneeriselle skriptiohjelmalle.

3.4.2 Raporttitiedoston lähettäminen

Kaupparaportin XML-dokumentti tulee lähettää Finanssivalvonnalle TYVI-raportointipalvelun kautta. Kaupparaportoinnin teknisen dokumentaation ”Kaupparaportoinnin konekielisen tietojenvälityksen kuvaus” luvussa 3.2 esitellään yhtenä vaihtoehtona raportin lähettäminen käyttäen FTP-tiedostosiirtoa. [4] Ohjelmallisesti FTP-tiedostosiirto on mahdollista toteuttaa kaupparaportoinnin osalta, koska tiedostosiirtoa edellyttävässä vaiheessa on olemassa valmiiksi generoitu XML-dokumentti binääridatana, joka saadaan ohjelmallisesti muodostettua minkä tahansa FTP-asiakasohjelman ymmärtämäksi lähetettäväksi tiedostoksi. Ohjelmallisesti tehtävää FTP-tiedostosiirtoa varten tarvitaan siirrettävän tiedoston lisäksi kohdepalvelimen osoite, kohdekansio sekä käyttäjätunnus ja salasana.

3.4.3 Vastaustiedoston noutaminen

Aiemmin luvussa 3.2 esitellyn mukaisesti lähetty kaupparaporttitiedosto prosessoidaan vastaanottajan päässä noin kymmenessä minuutissa, joten kyseisen ajanjakson jälkeen tulisi ohjelmallisesti toteutetun FTP-asiakasohjelman alkaa tehdä nouto-kyselyä TYVI-palvelun vastaustiedostoille varattuun tiedostohakemistoon. Vastaustiedoston löytyessä tulisi ohjelmallisesti toteutetun asiakasohjelman siirtää tiedosto FA Portfolio -sovelluksen palvelimelle tulkittavaksi.

Vastaustiedoston ollessa määrämuotoinen XML-dokumentti, joka noudattaa määritetyn skeeman mukaista sisältöä, voidaan se täten tulkita täysin ohjelmallisesti käyttäen mitä tahansa valmisohjelmakirjaston XML-parseria.

3.4.4 Tapahtumien merkkaus kaupparaportoiduiksi

Vastaustiedoston XML-dokumentti sisältää yksilöllisenä tietueina jokaisen aiemmin raportoitavaksi lähetetyn kauppatapahtuman statuksen ja tarkennekoodin. Raportoitavaksi lähetetyn kauppatapahtuman yhdistävänä tekijänä vastaustiedostolla olevaan statustietoon on tunnisteena toimiva TransactionReferenceNumber.

Kunkin raportoiduksi lähetetyn tapahtuman osalta tiedostolla on statustieto raportoinnista. Statuksia ovat hyväksytty (ACCEPTED), ohitettu (IGNORED) ja hylätty (FAILED). Tapahtuman statuksen ollessa ohitettu on samalla TransactionReferenceNumberilla jo aiemmin raportoitu kauppatapahtuma onnistuneesti. Hylättyjen tapahtumien osalta vastaustiedostolla on tarkennekoodi, jonka perusteella selviää hylkäyssyy. Taulukko 2:ssa on esitelty eri hylkäyssyiden tarkennekoodit ja niiden tarkoitukset.

Hyväksyttyjen tapahtumien osalta voidaan FA Portfolio -sovelluksen palvelurajapintoja käyttäen ladata ohjelmallisesti kukin tapahtuma TransactionReferenceNumberin perusteella, joka noudattaa syntaksia <sijiotussalkun_tunnus>-<tapahtumanumero>. Tällöin ensin ladataan sijoitussalkku arvolla, joka on ”-” -merkin vasemmalla puolen ja sitten tapahtuma kyseisestä sijoitussalkusta ”-” -merkin oikealla puolella olevalla arvolla. Tapahtuma-entiteetin löytyessä korvataan tapahtuman tagitieto uudella arvolla ”Kaupparaportointi – raportoitu” ja tallennetaan muokattu tapahtuma.

Taulukko 2. Mahdolliset hylkäyssyyt raportoidulle kauppatapahtumalle.

AIIDT	Johdannaistyypin (AIIDerivativeType) on oltava [O] tai [F]
AIIMD	Eräpäivän (AIIMaturityDate) on oltava kelvollinen päiväys
AIIPC	Indikaattorin (AIIPutCallIdentifier) on oltava [P],[C] tai [F]
AIISP	Lunastushinta (AIIStrikePrice) virheellinen
AIIXC	Tuotekoodia (AIIExchangeCode) ei löydy tai se on virheellinen
AIIXPC	Tuotekoodin (AIIExchangeProductCode) oltava 1-12 merkkinen
ICPC	Vastapuolitunnus on virheellinen tai se ei ole voimassa kauppapäivänä
IEXD_TD	Kauppatapahtuman kaupantekoaika pitää olla sama tai aikaisempi kuin instrumentin voimassaolon päättymispäivä
IISIN	Virheellinen ISIN-koodi
IPN	Valuuttakoodi on virheellinen tai se ei ole voimassa kauppapäivänä
IPNIT	Euroa edeltävää valuuttakoodia (ATS,BEF,CYP,DEM,ESP,FIM,FRF,GRD, IEP,ITL,LUF,MTL,NLG,PTE,SIT ja SKK) saa käyttää vain velkainstrumentille
IRFTD	Raportoijan BIC-koodi on virheellinen tai se ei ole voimassa kauppapäivänä
ITD	ITD Virheellinen kaupantekoaika
IVI	Kauppapaikan tunnus on virheellinen tai se ei ole voimassa kauppapäivänä
MTI	Puuttuva tapahtuman tunnus
CIM	Puuttuva asiakastieto
ICC	Asiakastunnus on virheellinen tai se ei ole voimassa kauppapäivänä
RTAC	Viitattu kauppatapahtuma on jo peruttu
VII	AII-kauppapaikalla kauppatapahtuman instrumentti on yksilöitävä AII-tunnuksella, muilla kauppapaikoilla ISIN-koodilla
IUUISIN	Ultimate Underlying ISINCode on väärän muotoinen
IPMV	Price multiplier pitää olla positiivinen
ISPV	Strike price pitää olla positiivinen silloin kun tieto on annettu

4 Apache Camel – sovelluskehys integraatioille

Apache Camel on Javalla toteutettu, avoimen lähdekoodin omaava sovelluskehys. Camel-sovelluskehys tarjoaa mahdollisuuden eri järjestelmien, tekniikoiden ja tietotyyppien integroitumisen toisiinsa sovelluskehyksen viestienvälityksen reititysmekanismilla. [6] Camelin kehitysprojekti alkoi vuonna 2007 ja sillä on aktiivinen kehittäjäyhteisö, joiden kehitystyö on nyt edennyt Camel-sovelluskehyksen versioon 2.14.4 [7]. FA Portfolio -sovelluksessa on toteutettu tuki Camel-sovelluskehykselle, jonka avulla on toteutettu lukuisia integraatioratkaisuja palvelemaan sovelluksen peruskäyttöä sekä asiakaskohtaisia, räätälöityjä integraatioratkaisuja FA Portfolio -sovelluksen ja asiakkaan muiden järjestelmien välille.

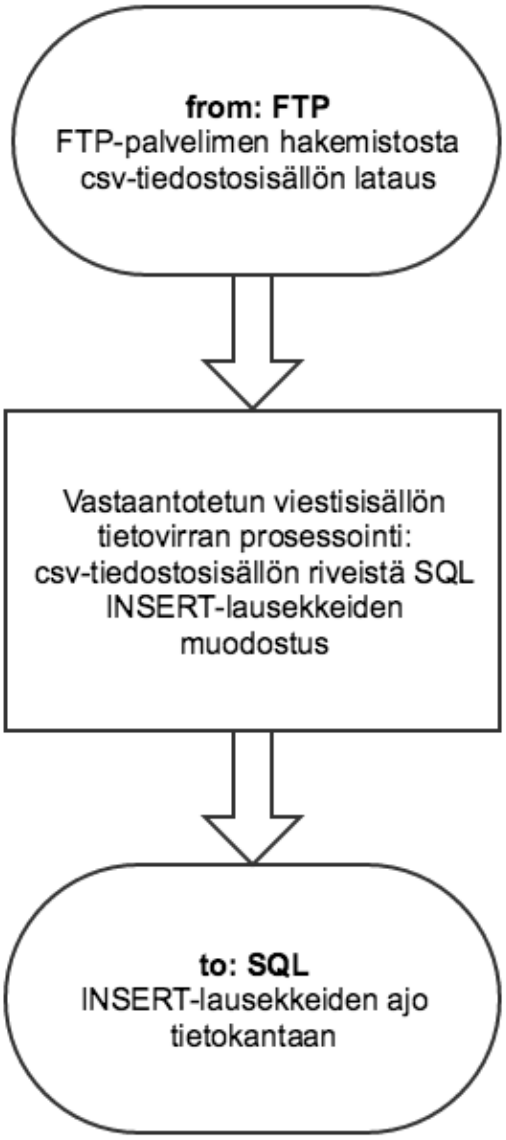
4.1 Viestienvälityksen reititysmekanismi

Camelin ytimenä toimii viestienvälityksen reititysmekanismi, joka mahdollistaa sovelluskehyksen käyttäjän määrittää haluamansa reitityssäännöt. Reitityssäännön määritys koostuu kolmesta osasta: mistä lähteestä viesti hyväksytään vastaanotettavaksi, miten vastaanotettu viesti prosessoidaan sekä prosessoidun viestin mahdollisesta uudesta vastaanottajasta.

Kuvassa 8 on havainnollistettuna integraatioreititys, jossa FTP-palvelimen hakemistossa olevan CSV-tiedoston tietosisällön perusteella luodaan SQL-tietokantapalvelimen tauluun uusia rivejä INSERT-lausekkeiden avulla.

Reititysintegraation ensimmäinen tehtävä on kirjautua ftp-palvelimelle ja noutaa määritetystä hakemistosta CSV-tiedoston sisältö *java.io.FileInputStream*-tyyppisenä tietovirtana reititysintegraatiolle prosessoitavaksi.

Prosessointivaiheessa vastaanotettu CSV-tiedoston tietosisältö luetaan sarake ja rivi kerrallaan ja näistä muodostetaan SQL-kielen mukaiset INSERT-lausekkeet tietokantataululle. Reitityksen seuraavalle vaiheelle INSERT-lausekkeet välitetään *java.lang.String*-tyyppisessä tietovirrassa, jotka tullaan ajamaan tietokantapalvelimen tietokantatauluun.



Kuva 8. Camel-sovelluskehyksen integraatioreititys CSV-tiedoston tietosisällön ajaminen tietokantatauluun.

Camel-sovelluskehys tarjoaa abstraktin ohjelmointirajapinnan viestisisällön käsittelylle. Sovelluskehyksen ohjelmointirajapinta ei oleta vastaanotetun tai eteenpäin lähetettävän viestisisällön olevan tietyn tietotyypin mukaista tietovirtaa. Tämä mahdollistaa sovelluskehyksen käyttäjän toteuttaa integraatioreitityksiä minkä tahansa järjestelmien välillä, kunhan käyttäjä vain pitää huolen, että tietovirran vastaanottaja ymmärtää tietovirran sisällön.

4.2 Camelin viestintämalli

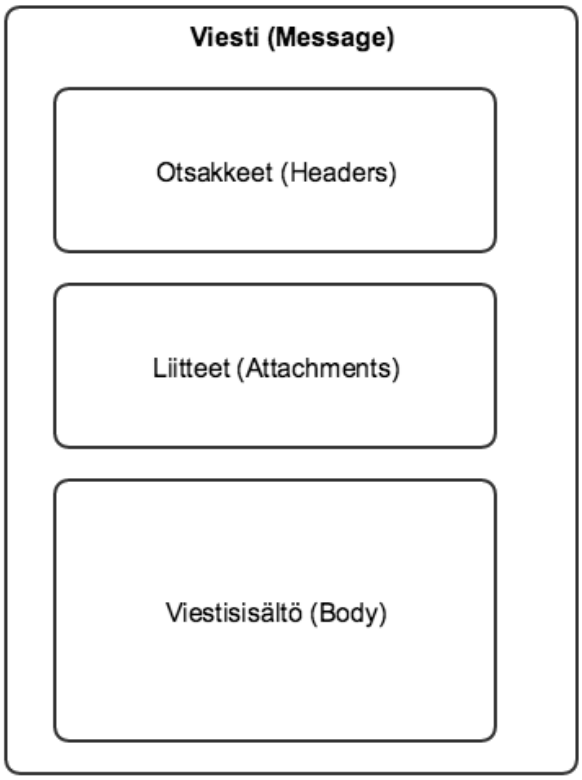
Camelin viestintämalli koostuu kahdesta abstraktista entiteetistä:

- *org.apache.camel.Message* - Viesti jota ollaan välittämässä. Sisältää muun muassa reititettävän tietosisällön Camel-sovelluskehyksessä.

- *org.apache.camel.Exchange* - Välitys, jossa viestinvälitys tapahtuu. Exchange-entiteetin tulee aina sisältää sisään tulevan viesti-entiteetin (in Message) ja integraatioreittikohtaisesti myös mahdollisen eteenpäin välitettävän viesti-entiteetin (out Message).

4.2.1 Viesti

Viesti (Message) on kommunikoinnin väline, jolla on sisältö, lähettäjä ja vastaanottaja. Camel-sovelluskehyksessä viesti liikkuu reititysmekanismin avulla viestin lähettäjältä viestin vastaanottajalle. Kuvan 9 mukaan Camelin viesti voi koostua otsakkeista (Headers), liitteistä (Attachments) ja viestisisällöstä (Body).



Kuva 9. Camelin viesti-entiteetin koostumus.

Otsakkeet ovat Map-tietorakenteen sisällä olevia avain-arvo-pareja, jotka kuvailevat viestiä, esimerkiksi tietoa lähettäjästä tai viestisisällön tietotyypistä. Avain on yksilöllinen tietotyypiltään *java.lang.String* oleva tieto ja arvo tietotyyppiriippumaton *java.lang.Object*-arvo.

Camel-viestiin on myös mahdollisuus sisällyttää liitteitä. Liitteitä käytetään esimerkiksi integraatioreitissä, jonka lähettäjänä tai vastaanottajana on sähköpostiviesti, jossa sähköpostiviestin liitetiedoston binääridata asetetaan Camel-viestin liiteosioon.

Camelin viestisisältö on tietotyyppiriippumaton *java.lang.Object*-arvo. Tämä mahdollistaa viestisisältöjen vastaanottamisen prosessoitavaksi mistä tahansa järjestelmästä, riippumatta viestisisällön tietotyypistä. Viestin lähettäjän ja vastaanottajan käyttäessä eri tietotyyppejä - viestin muodostuksessa lähettäjän päässä ja viestin tulkitsemisessa vastaanottajan päässä. Tällöin integraatioreitin suunnittelijalle/toteuttajalle jää vastuu siitä, että vastaanotettu viesti osataan välittää eteenpäin ja että vastaanottaja ymmärtää välitetyn viestin sisällön. Camel-sovelluskehys tarjoaakin tähän käyttäjää helpottamaan mekanismin lukuisien eri tietotyyppien automaattiseen tyyppimuunnokseen tapauskohtaisesti ymmärrettävään muotoon.

4.2.2 Välitys

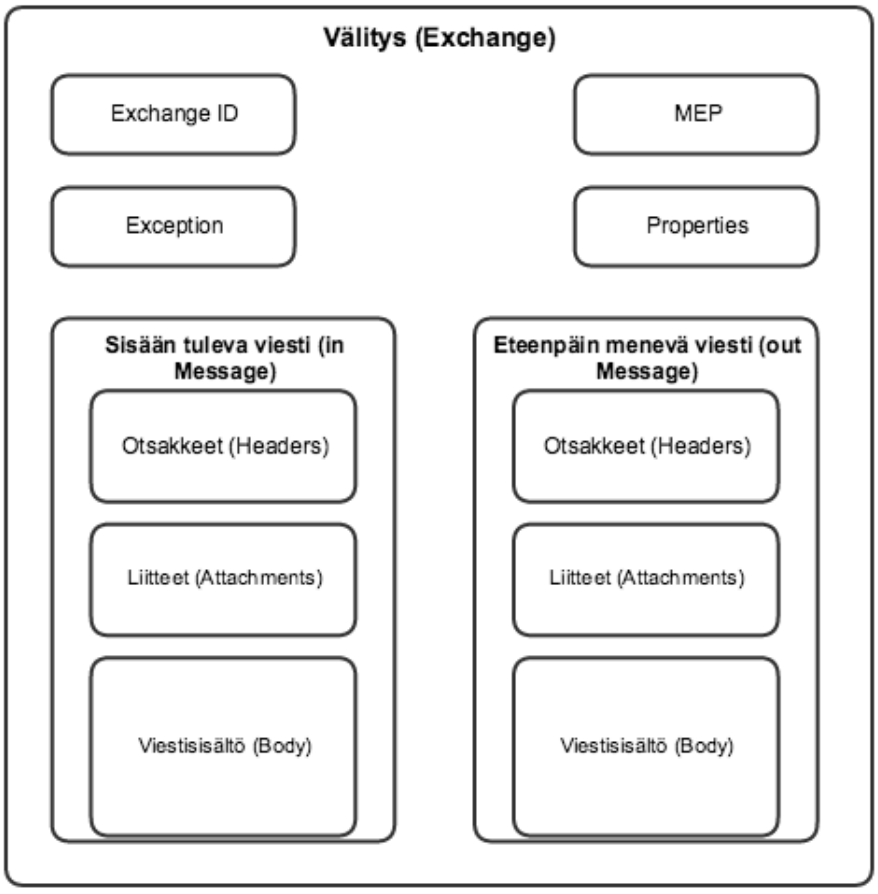
Välitys (Exchange) on Camel-viestin säiliö (container) integraatioreitityksen ajan. Välitys-entiteetti yksilöidään uniikilla Exchange ID -arvolla erottumaan muista mahdollisista Camel-sovelluskehyksessä olevista integraatioreittien välitys-entiteeteistä.

Välitys-entiteetti sisältää sisään tulevan viesti-entiteetin (in Message) ja mahdollisen eteenpäin menevän viesti-entiteetin (out Message).

Välitys-entiteetti sisältää myös tietona, minkä tyyppistä viestinvälitysmallia MEPs (Message Exchange Patterns) integraatioreitti käyttää. Välitys-entiteetillä on käytettävissä joko yksisuuntainen InOnly-viestinvälitysmalli tai kutsu-vastaus-tyyppinen InOut-viestinvälitysmalli.

Kuvassa 8 esitetty integraatioreitti käyttää välityksessä InOut-tyyppistä viestinvälitysmallia, jossa sisään tulevan viesti-entiteetin viestisisältö koostuu CSV-tiedoston tietovirrasta (java.io.FileInputStream). Prosessoinnin jälkeen eteenpäin menevä viesti-entiteetin viestisisältö on SQL-kielen INSERT-lausekkeet sisältävä merkkijono (java.lang.String).

Välitys-entiteetillä on lisäksi käytössä Exception- ja Properties-kentät. Exception-kenttään talletetaan integraatioreitin aikana mahdollisesti tulevat virheet ja poikkeukset, jotka ovat integraatioreitityksen toteuttajalla mahdollista käsitellä haluamallaan tavalla. Properties-kentässä on mahdollista pitää yllä avain-arvopareittain sovelluskehykselle globaaleja asetuksia. Kuvassa 10 on havainnollistettuna välitys-entiteetin koostumus.



Kuva 10. Camelin välitys-entiteetin koostumus

4.3 Camel-integraatioreitti

Camel-integraatioreitti koostuu yhdestä tai useammasta päätepisteestä (Endpoint). Päätepisteitä on kahdentyyppisiä: kuluttaja-tyyppisiä (Consumer) sekä tuottaja-tyyppisiä (Producer).

Integraatioreitin käynnistävä *from*-alkuinen päätepiste on aina tuottaja-tyyppinen, joka tuottaa integraatioreitin välitys-entiteetille viestin prosessoitavaksi. Yhdellä integraatioreitillä täytyy olla ainoastaan yksi tuottaja-tyyppinen päätepiste, jonka tehtävä on siis käynnistää integraatioreitti ja tuottaa reitillä välitettävä viesti.

Kuluttaja-tyyppisiä päätepisteitä ovat integraatioreitillä määritetyt *to*-alkuiset päätepisteet. Kuluttaja-tyyppiset päätepisteet vastaanottavat tuottajan alullepaneman viestin ja käyttävät viestisisältöä omien tarkoitusperiensä mukaisesti. Integraatioreitillä voi olla useampi kuluttaja-tyyppinen päätepiste käyttämässä reitillä välitettävää viestisisältöä.

4.4 Camelin valmiskomponentit integraatioille

Valmiskomponentit ovat lisämoduuleita Camel-sovelluskehykselle, joka mahdollistavat lukuisien eri järjestelmien, protokollien ja tietotyyppien integroitumisen toisiinsa sovelluskehyksen reititysmekanismin avulla. Valmiskomponentti-moduuleita löytyy yli 80 kappaletta erilaisiin integraatiotarpeisiin. [6] Kattava listaus komponenteista, niiden käyttötarkoituksista sekä dokumentaation komponenttien käytöstä löytyy Apache Camel -sovelluskehyksen kotisivuilta [8].

Komponentteja käytetään integraatioreitillä URI-notaatiomääreillä, joka noudattaa syntaksia *skeema:konteksti?lisäasetukset.* Skeema kuvaa käytettävän komponentin nimeä. Kontekstilla kerrotaan, missä tai minkälaisessa yhteydessä komponenttia käytetään. Lisäasetukset ovat komponenttikohtaisia lisämääreitä, joilla voidaan vaikuttaa komponentin toimintaan ja sen tuottamaan lopputulokseen.

Koodiesimerkissä 1 on esitettynä luvussa 4.1 kuvattu esimerkki integraatioreitistä, jossa käytetään FTP- ja SQL-komponentteja CSV-tiedoston tietosisällön kirjoittamiseen SQL-

palvelimen tietokantatauluun. Komponenttien URI-määreet annetaan Camel-sovelluskehyksen tukemalla XML-kieleen pohjautuvalla Spring DSL -kielellä.

```
<route>
    <from uri="ftp://ftp.server.fi/tilaukset?username=user&password=secret&move=.done"/>

    <setBody>
        <groovy>
            Groovy-skriptillä toteuttu ohjelmallinen käsittely,
            jossa muodostetaan vastaanotetusta FileInputStream:sta
            eteenpäin menevän out Messagen INSERT-lausekeet String-
            tyyppiseksi viestisisällöksi.
        </groovy>
    </setBody>

    <to uri="sql:{out.body}?dataSource=myDataSourceBean"/>

</route>
```

Koodiesimerkki 1. Esimerkki valmiskomponenttien käyttämisestä Spring DSL -kielellä integraatioreitillä.

Koodiesimerkissä 1 esiteltynä oleva integraatioreitti käynnistyy tuottaja-tyyppisenä skeemassa esitellyllä FTP-komponentilla. Kontekstissa kerrotaan FTP-palvelimen osoite ja kohdehakemisto nimeltään /tilaukset. Lisäasetuksissa kerrotaan millä käyttäjätunnuksella ja salasanalla kirjautuminen toteutetaan. Hakemiston sisältäessä tiedoston, noudetaan tiedoston tietosisältö prosessoitavaksi ja niin ikään lisäasetusten mukaan siirretään käsitelty tiedosto FTP-palvelimella hakemistoon .done, jotta samaisen tiedoston tietosisältöä ei enää toistamiseen noudettaisi.

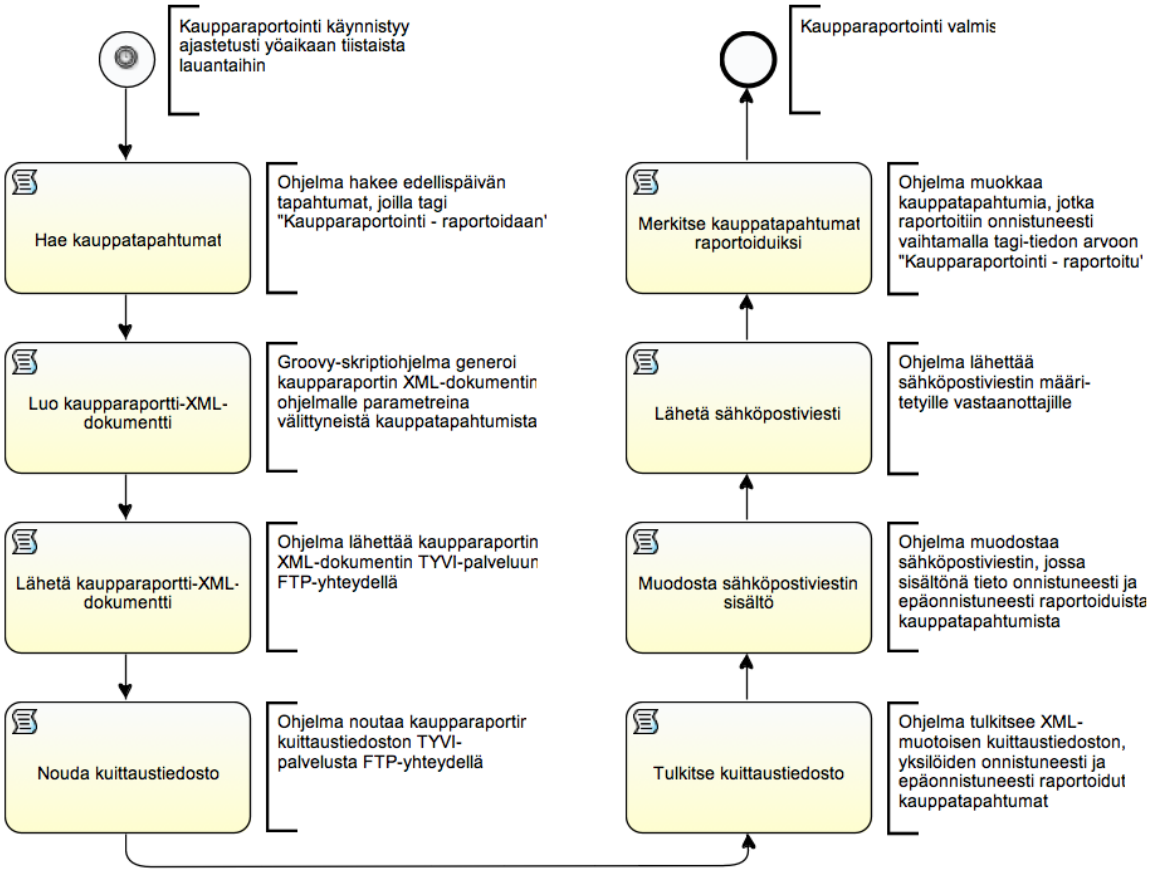
Kuluttaja-tyyppiselle päätepisteelle eteenpäin menevä out Messagen viestisisältö tuotetaan ohjelmallisesti käyttäen Groovy-skriptiä prosessointivaiheessa. Viestisisältö välitetään kuluttajana toimivalle SQL-komponentille, joka kontekstissaan käyttää out Messagen viestisisällössä olevaa SQL-kielen INSERT-lausekkeita lisätäkseen rivejä tietokantatauluun. Lisäasetuksena SQL-komponentille kerrotaan sovelluskehykselle esitellyn JDBC-tyyppisen tietokantayhteyden tiedot sisältävän entiteetin nimi.

5 Kaupparaportoinnin automatisointi

Automatisoitu kaupparaportointi tulisi luvussa 3.3 esitetyn suunnitelman mukaisesti pystyä ohjelmallisesti käynnistymään itsestään, tuottamaan kaupparaportti, lähettämään raportti TYVI-palveluun, noutamaan vastaustiedosto TYVI-palvelusta, tulkitsemaan

vastaustiedoston sisältö ja merkitsemään kauppatapahtumat raportoiduksi. Lisäksi FA Portfolio -sovelluksen käyttäjää tulee informoida sähköpostitse onnistuneesti ja epäonnistuneesti raportoitujen kauppatapahtumien osalta.

Kuvassa 11 on esitelty ohjelmalliset, vaiheistetut tehtävät, jotka yhdessä muodostavat automatisoidun kaupparaportoinnin.



Kuva 11. Automatisoidun kaupparaportoinnin ohjelmalliset vaiheet.

5.1 Kaupparaportoinnin käynnistys

Automatisoidun kaupparaportoinnin ensimmäisen integraatioreitin tuottaja-tyyppisen päätepisteen tehtävänä on ajastetusti käynnistää integraatioreitti. Camel-sovelluskehyksen valmiskomponenteista tähän tarkoitukseen löytyy Quartz-komponentti. Quartz-komponentilla pystytään luomaan ajastin, jolle käynnistysaika määritellään Unix-pohjaisista käyttöjärjestelmistä tutulla cron-syntaksilla. Cron-syntaksin

avulla on mahdollista määrittää kaupparaportoinnille tarpeen oleva käynnistysfrekvenssi: tietty kellonaika vuorokaudessa. Viikonpäivät ovat tiistaista lauantaihin.

5.2 Kaupparaportin muodostus

Integraatioreitti käynnistyi tuottaja-tyyppisenä päätepisteenä toimineen Quartz-komponentin toimesta, joka loi Camel-viestin välitys-entiteetille. Välitys-entiteetillä olevan Camel-viestin viestisisältö (Body) on tyhjä, koska Quartz-komponentti lisää ainoastaan tietoja viestin otsakkeisiin (Headers). Nämä tiedot eivät ole tässä tapauksessa integraatioreitillä oleellisia.

Kaupparaportoinnin käynnistyksen jälkeen tulisi seuraavaksi määritelmän mukaan hakea FA Portfolio -sovelluksesta edellispäivän raportoitavat tapahtuma-entiteetit ja muodostaa niiden pohjalta kaupparaportin XML-dokumentti. Integraatioreitin viestisisällön ollessa tyhjä voidaan viestisisältö muodostaa integraatioreitin prosessointivaiheessa täysin halutulla tavalla. Luvussa 3.2 kerrottiin FA Portfolio -sovellukselle toteutetusta Groovy-skriptiohjelmasta, joka muodostaa kaupparaportin XML-dokumentin. Integraatioreitin prosessointivaiheessa voidaan käyttää kyseistä skriptiohjelmaa laajentaen sitä vain siten, että skriptiohjelma hakee ensin edellispäivän kaupparaportoitaviksi merkityt tapahtuma-entiteetit.

Camel-sovelluskehyksen valmiskomponentin Language avulla on mahdollista päästä ohjelmallisesti sekä lukemaan että muokkaamaan integraatioreitin välitys- ja viesti-entiteettien sisältöä. Language-komponentti tukee monia skriptikieliä, joista yhtenä on myös FA Portfolio -sovelluksessa laajasti käytetty Groovy-kieli. [9]

Tässä tapauksessa on siis tarpeen muokata Groovylla viesti-entiteettiä siten, että ensin haetaan tarvittavat tapahtuma-entiteetit käyttäen FA Portfolio -sovelluksen palvelurajapintoja. Tapahtuma-entiteeteistä muodostetaan aiemmin toteutetun skriptiohjelman mukaisesti kaupparaportin sisältävä XML-dokumentti - *java.io.FileOutputStream*-tyyppisenä tietovirtana ja asetetaan se eteenpäin menevän viestin viestisisällöksi. Eteenpäin menevän viestin otsakkeisiin asetetaan kaupparaportin tiedostonimeksi yhdistelmä, joka koostuu Finanssivalvonnan teknisen dokumentaation mukaan syntaksista TR_<sijoituspalveluyrityksen BIC-tunniste>_<vvvvkkpp>_0001.XML [4].

Integraatioreitin kuluttajatyyppisenä päätepisteenä toimii File-komponentti, joka lukee Groovy:lla prosessointivaiheessa muodostetun viestisisällön ja luo sen pohjalta tiedoston palvelimen tiedostojärjestelmään. Luotavan tiedostonimen File-komponentti saa ymmärtämästänsä viestin otsakkeesta CamelFileName, joka alustettiin viestin prosessointivaiheessa sisältämään kaupparaportin nimi.

Kaupparaportoinnin automatisoinnista on täten toteutettuna koodiesimerkin 2 mukainen integraatioreitti, jossa ajastetusti käynnistetään raportointi ja tallennetaan kaupparaportin sisältävä XML-dokumentti palvelimen tiedostojärjestelmään FTP-lähetystä varten.

```
<route>
    <from uri="quartz://timerName?cron=0+0+2+*+*+TUE-SAT"/>

    <setBody>
        <groovy>
            Groovy-skriptillä muokataan eteenpäin välitettävää viesti-entiteettiä:
            - Haetaan edellispäivän raportoitavat kauppatapahtumat
            - Muodostetaan kaupparaportti FA Portfolio -sovelluksen
              Groovy-skriptiohjelmalla ja asetetaan xml-dokumentin sisältävä
              FileOutputStream viestin viestisisällöksi
            - Asetetaan kaupparaportin tiedostonimi viestin otsakkeisiin
        </groovy>
    </setBody>

    <to uri="file:/mnt/kaupparaportointi/lahtevat"/>

</route>
```

Koodiesimerkki 2. Automatisoidun kaupparaportoinnin ensimmäinen integraatioreitti Spring DSL -kielellä kuvattuna.

5.3 Kaupparaportin lähetys ja vastaustiedoston noutaminen

Kaupparaportin sisältämän XML-tiedoston lähetys TYVI-palveluun käyttäen tiedonsiirrossa FTP-protokollaa on yksi suoraviivaisimmista Camel-sovelluskehyksellä tehtävistä integraatioreiteistä.

Integraatioreitti käynnistetään File-komponentilla, joka havaitsee määritetyssä hakemistossa olevan uuden tiedoston. Integraatioreitille luodaan välitys-entiteetti, jonka viestin viestisisältönä on tiedoston binääridata tietovirtana ja viestin otsakkeissa tietona muun muassa tiedoston nimi, tyyppi ja koko.

Integraatioreitillä eteenpäin välitettävä viesti kelpaa FTP-komponentille sellaisenaan ilman, että viestiä tarvitsisi sen kummemmin prosessoida. Tästä muodostetaan koodiesimerkin 3 mukainen integraatioreitti tiedoston siirtämiseksi palvelimen tiedostojärjestelmästä TYVI-palvelun FTP-hakemistoon.

```
<route>
    <from uri="file:/mnt/kaupparaportointi/lahtevat?move=.done"/>

    <to uri="ftp://tyvi.palvelun.osoite/tyvi/data?username=user&password=secret"/>
</route>
```

Koodiesimerkki 3. Kaupparaportin sisältävän XML-tiedoston lähetys FTP:llä TYVI-palveluun.

Integraatioreitillä käytettävälle File-komponentille kerrotaan kontekstissa hakemisto, joka on sama, jonne ensimmäinen integraatioreitti XML-tiedoston loi. Lisäasetuksena komponentille kerrotaan, että se siirtää käsitellyn tiedoston .done-hakemistoon, jotta samaista tiedosta ei enää uudelleen lähetettäisi.

Integraatioreitin viestin kuluttaja-tyyppinä toimivan FTP-komponentin kontekstissa kerrotaan FTP-palvelimen osoite ja tiedostosiirron kohdehakemisto ja lisäasetuksissa FTP-yhteyttä varten käytettävä käyttäjätunnus ja salasana.

TYVI-palvelusta saatavan vastausviestin noutamiseen tarvittava integraatioreitti hoituu koodiesimerkin 4 mukaisesti samojen komponenttien avulla kuin kaupparaportin XML-tiedoston siirtäminenkin. Ainoastaan komponenttien päätepisteiden järjestys on päinvastainen. Integraatioreitti käynnistetään FTP-komponentilla, joka muodostaa ja pitää yllä FTP-yhteyttä TYVI-palvelun FTP-palvelimelle. FTP-yhteyden jostain syystä katketessa yrittää integraatioreitti välittömästi muodostaa uuden yhteyden. Vastausviestin valmistuessa FTP-palvelimen hakemistoon, siirretään vastausviestin tietosisältö File-komponentille, joka kirjoittaa integraatioreitillä välitettävän viestin viestisisällön FA Portfolio -palvelimen tiedostojärjestelmään.

```
<route>
    <from uri="ftp://tyvi.palvelun.osoite/response?username=user&password=secret&move=.done"/>

    <to uri="file:/mnt/kaupparaportointi/vastausviestit"/>
</route>
```

Koodiesimerkki 4. Vastausviestin nouto FTP:llä ja tiedoston tallennus palvelimen tiedostojärjestelmään.

5.4 Vastausviestin tulkitseminen ja sähköpostiviestin muodostus

Vastausviestin tulkitsemiseen ja sähköpostiviestin muodostamiseen liittyvä integraatioreitti käynnistyy File-komponentin avulla. Komponentti havaitsee kontekstilleen kerrotussa palvelimen tiedostohakemistossa uuden XML-muotoisen vastausviestitiedoston ja välittää tiedoston tietovirran eteenpäin prosessoitavaksi koodiesimerkissä 5 esitetyn mukaisesti.

```
<route>
    <from uri="file:/mnt/kaupparaportointi/vastausviestit?move=.done"/>

    <multicast>
        <to uri="xslt:file:/mnt/kaupparaportointi/templates/sahkopostiviesti.xslt"/>

        <to uri="direct:merkitseTapahtumatRaportoiduiksi"/>
    </multicast>

</route>
```

Koodiesimerkki 5. XML-muotoisen vastausviestin sisällön välitys eteenpäin prosessoitavaksi.

Integraatioreitin koodiesimerkissä 5 on käytetty Camel-sovelluskehyksen multicast-välitysominaisuutta, jonka avulla välitettävän viestin sisältö voidaan lähettää eteenpäin useammalle kuluttajatyyppiselle päätepisteelle.

Sähköpostiviestin sisältö muodostetaan kuluttajatyyppisen XSLT-valmiskomponentin avulla. Komponentti ottaa viestinä vastaan XML-dokumentin tietovirran, joka prosessoidaan komponentilla käyttäen määritettyä xslt-tyylitiedostoa, joka tulee muodostaman html-tyyypisen tiedoston palvelimen tiedostojärjestelmään. Komponentin tukeman XSLT-muunnoskielen avulla xslt-tyylitiedostossa haetaan XML-muotoisen vastausviestin rakenteesta elementtien arvoja, joista muodostetaan html-taulukolle sisältä. Taulukon rivitietona on raportoitujen kauppojen TransactionReferenceNumber, statustieto joko hyväksytty tai hylätty sekä hylättyjen kauppojen osalta hylkäyssyy. Prosessointivaiheen päätteeksi palvelimen tiedostojärjestelmään muodostetaan html-tiedosto, jonka tietosisältö tulee toimimaan sähköpostiviestin sisältönä.

5.5 Sähköpostiviestin lähetys

Sähköpostiviestin lähettävä integraatioreitti käynnistetään File-komponentin avulla. Komponentti havaitsee kontekstissaan määritetyssä tiedostohakemistossa luvussa 5.4 esitellyn mukaisesti luodun html-tiedoston, jonka tietovirran File-komponentti välittää eteenpäin prosessoitavaksi ja siirtää html-tiedoston .done-hakemistoon. Viestin tietosisältö välitetään koodiesimerkin 6 mukaisesti kuluttajatyyppiselle SMTP-komponentille.

```
<route>
    <from uri="file:/mnt/kaupparaportointi/sahkopostiviestit?move=.done"/>

    <setHeader headerName="subject">
        <simple>Kaupparaportoinnin vastaussanoma</simple>
    </setHeader>

    <setHeader headerName="Content-Type">
        <simple>text/html</simple>
    </setHeader>

    <to uri="smtp://smtp.server.fi?to=backoffice@sijoituspalveluyritys.fi&from=fa.portfolio@fasolutions.fi"/>
<route>
```

Koodiesimerkki 6. HTML-tyyppisen tiedoston sisällön lähettäminen sähköpostitse.

Integraatioreitillä asetetaan eteenpäin lähtevälle viestille otsakkeita, jotka vastaanottajana toimiva SMTP-komponentti osaa tulkita. Otsakkeissa annetaan sähköpostiviestille otsikko (subject) sekä kerrotaan sähköpostin viestisisällön (Content-Type) olevan html-tekstiä.

SMTP-komponentin kontekstissa kerrotaan sähköpostipalvelimen osoite. Lisäasetuksissa komponentille kerrotaan viestin vastaanottajan sähköpostiosoite *to*-määreellä sekä sähköpostiviestin lähettäjä *from*-määreellä.

5.6 Kauppatapahtumien merkitseminen raportoiduiksi

Kauppatapahtumien merkitseminen raportoiduksi tapahtuu integraatioreitissä, joka on viestinvälitysmalliltaan *InOnly,* koska se vain vastaanottaa viestin prosessoitavakseen eikä välitä sitä enää eteenpäin. Integraatioreitin käynnistää Direct-valmiskomponentti, joka on Camel-sovelluskehyksen sisäinen viestinvälittäjä. Direct-komponentin avulla on mahdollista välittää käsiteltävä viesti integraatioreitiltä toiselle Camel-integraatioreitille. [11]

Koodiesimerkeissä 5 esitelty integraatioreitti välitti XML-dokumentin tietovirran sisällön luvussa 5.4 esitellylle XSLT-komponentille sekä Direct-komponentille, joka käynnistää koodiesimerkin 7 mukaisen integraatioreitin.

```
<route>
    <from uri="direct:merkitseTapahtumatRaportoiduiksi"/>

    <process>
        <groovy>
            Groovy-skriptissä parsitaan XML-muotoisen vastausviestin
            sisältö käyttäen groovy.util.XmlParser-luokan parsimiskeinoja.

            Parsinnan jälkeen skriptiohjelmalla haetaan TransactionReference-
            Numbereita vastaavat kauppatapahtumat FA Portfolio -sovelluksen
            rajapintapalveluita käyttäen ja asetetaan tagi "Kaupparaporointi -
            raportoitu" tapahtumille, jotka raportoitiin onnistuneesti statuksella
            ACCEPTED.
        </groovy>
    </process>

</route>
```

Koodiesimerkki 7. Integraatioreitti, joka prosessoi XML-dokumentin tietosisällön ja merkitsee skriptiohjelman avulla kauppatapahtumat raportoiduiksi.

Integraatioreitille välitettävä vastausviestin XML-dokumentin tietosisältö prosessoidaan Camel-sovelluskehyksen Language-komponentin tukemalla Groovy-skriptiohjelmalla, jossa käytetään XML-dokumentin parsintaan tarkoitettua *groovy.util.XmlParser*-luokkaa. Parsijan avulla saadaan XML-dokumentista eriteltyä raportoidut kauppatapahtumat, joilla statustietona on ACCEPTED. Vastausviestissä kauppatapahtuma on yksilöitynä TransactionReferenceNumber-arvolla, jonka perusteella skriptiohjelman on mahdollista hakea sitä vastaava tapahtuma-entiteetti FA Portfolio -sovelluksen palvelurajapintaa käyttäen. Onnistuneesti raportoiduksi merkittyjen TransactionReferenceNumberien perusteella tapahtuma-entiteetit haetaan muokattavaksi ja niille vaihdetaan tagi-tieto arvoon ”Kaupparaportointi - raportoitu”.

6 Yhteenveto

Opinnäytetyön tavoitteena oli jatkokehittää FA Portfolio -sovelluksen kaupparaportointiominaisuutta toimimaan mahdollisimman automatisoidusti. Kaupparaportin ollessa sovellusta käyttäville sijoituspalveluyrityksille pakollinen,

päivittäin Finanssivalvonnalle toimitettava raportti, haluttiin raportoinnin helpottamiseksi tarjota uutena lisäominaisuutena käyttäjiä helpottava automatisoitu raportointiversio.

Opinnäytetyössä kerrottiin yleisellä tasolla salkunhallintajärjestelmistä ja esiteltiin FA Solutions Oy:n kehittämä FA Portfolio salkunhallintaohjelmisto ja sen yksi tärkeimmistä elementeistä eli kauppatapahtuma, joiden perusteella rakentuu sijoitussalkkujen sisältö.

Kaupparaportoinnin automatisointia suunniteltaessa käytiin läpi vaiheittain kohdat, jotka aiemman käyttäjän vaatiman panoksen sijaan pystyttäisiin hoitamaan ohjelmallisesti. Automatisoitavat vaiheet päätettiin toteuttaa FA Portfolio -sovelluksen jo aiemmin tukemalla Apache Camel -sovelluskehyksellä. Opinnäytetyössä esiteltiin integraatioihin tarkoitetun Apache Camel -sovelluskehyksen pääperiaatteet: viestienvälityksen reititysmekanismi, viesti- ja välitys-entiteetit sekä sovelluskehykselle eri käyttötarpeisiin toteutettujen valmiskomponenttien käyttö.

Kaupparaportoinnin automatisointi käytiin läpi koodiesimerkein, jotka kuvasivat mitä kullakin integraatioreitillä tehtiin. Kaupparaportoinnin automatisointi koostui lopulta kuudesta integraatioreitistä, joiden avulla saavutettiin täysin automatisoitu kaupparaportointi koostuen: raporttitiedoston muodostamisesta, raportti- ja vastaustiedoston tiedonsiirrosta TYVI-palvelun ja sovelluspalvelimen välillä, vastaustiedoston tulkitseminen lähetettäväksi sähköpostiviestiksi ja onnistuneesti raportoitujen kauppatapahtumien merkitseminen raportoiduksi FA Portfolio -sovelluksessa.

Onnistuneen kaupparaportoinnin automatisoinnin avulla on FA Portfolio -sovellusta käyttäville sijoituspalveluyrityksille mahdollista tarjota käyttäjiä helpottava lisäominaisuus, joka on houkutteleva ominaisuus varsinkin sijoituspalveluyrityksille, jotka tekevät päivittäin yli kymmenen raportoitavaa kauppatapahtumaa.

Lähteet

1 Finanssivalvonta, Kaupparaportointi – Määräykset ja ohjeet 12/2013. Verkkodokumentti. Viitattu 11.10.2014. <http://www.finanssivalvonta.fi/fi/Saantely/Maarayskokoelma/Uusi/Documents/12_2013.M2.pdf>..

2 Vaadin, Book of Vaadin. Verkkodokumentti. Viitattu 5.9.2015. <https://vaadin.com/book/-/page/intro.html>.

3 Drools, Drools Documentation. Verkkodokumentti. Viitattu 4.10.2015. <https://docs.jboss.org/drools/release/6.2.0.Final/drools-docs/html_single/#d0e7259>.

4 Finanssivalvonta, Kaupparaportoinnin konekielisen tietojenvälityksen kuvaus. Verkkodokumentti Viitattu 31.10.2015. <http://www.finanssivalvonta.fi/fi/Raportointi/Raportointisovellukset/Kaupparaportointi/Documents/12_2013_konekielisen_tietojenvalityksen_kuvaus.pdf>.

5 Groovy, Creating XML. Verkkodokumentti. Viitattu 2.11.2015. <http://www.groovy-lang.org/processing-xml.html#_creating_xml>.

6 Manning Publications Co, Camel in Action. 2011. Claus Ibsen & Jonathan Anstey.

7 Apache Camel, Index. Verkkodokumentti. Viitattu 14.11.2015. <http://camel.apache.org/index.html>.

8 Apache Camel - Components. Verkkodokumentti. Viitattu 20.11.2015. <http://camel.apache.org/components.html>.

9 Apache Camel - Scripting Languages. Verkkodokumentti. Viitattu 21.11.2015. <http://camel.apache.org/scripting-languages.html>.

10 Apache Camel - File Component. Verkkodokumentti. Viitattu 21.11.2015. <http://camel.apache.org/file2.html>.

11 Apache Camel - Direct Component. Verkkodokumentti. Viitattu 29.11.2015. <http://camel.apache.org/direct.html>.